From Physical to Synthetic: ML-Driven Emulation of PUF Characteristics

Obeeda Al-Doulat, and Dr. Enas Abulibdeh

***Abstract*—Physically Unclonable Function (PUF) is an emerging technology that provides genuine security solutions for blockchain, integrated circuit (IC) trust, zero-trust architectures, and lightweight security systems. Silicon PUFs take advantage of the inherent physical variations of the underlying silicon [1] to generate a random stream (response), which can be used as a unique key for cryptographic systems or as a unique identifier for system authentication. However, the process of PUF design, implementation, verification, and security validation is costly and time-consuming, requiring substantial effort to analyze inputs and outputs, or Challenge-Response Pairs (CRPs). This work proposes using machine learning (ML) models to learn the physical characteristics of a PUF device and to generate synthetic CRPs that assist experts in analyzing, verifying, and validating PUF architectures at early stages of fabrication with minimal cost. Unlike ML-based modeling attacks, the proposed approach emulates the physical characteristics of the device rather than generating the exact response for a particular input (challenge). Consequently, the overall extracted CRPs from the real device exhibit characteristics similar to those of the synthetic CRPs. This study employs three different machine learning models: Conditional Variational Autoencoder (CVAE), Conditional Generative Adversarial Network (CGAN), and Multi-Layer Perceptron (MLP). CRPs extracted from an Arbiter PUF are used to train the models, and the performance of each model is evaluated by validating the synthetic CRPs using NIST 800-22 tests and by measuring the similarity between the real and synthetic CRPs.**

***Index Terms*—PUF, Machine Leaning, VAE, GAN, MLP, CRPs, synthesis, NIST 800-22, similarity.**

I. Introduction

PHysical Unclonable Functions (PUFs) are security primitive blocks that generate random responses based on the inherent physical variations of devices. To guarantee the randomness of these responses, the source of variation on the silicon chip, the response extraction approach, and the security validation process all depend on analyzing the real responses extracted from fabricated devices, where the physical characteristics are unclonable. However, the validation process of PUF devices is typically an open-loop process, where real responses are evaluated only after fabrication across multiple devices of the same architecture. In this scenario, improving the design and enhancing the quality of the responses is not feasible. Therefore, the PUF validation process is costly and requires multiple fabrication shuttles to validate a single architecture using the same technology through a specific supply chain. This work proposes a solution that provides experts with a synthetic CRP dataset generated by trained generative machine learning models, which can be utilized for architecture validation and evaluation. The proposed models: Conditional Variational Autoencoder (CVAE), Conditional Generative Adversarial Network (CGAN), and Multi-Layer Perceptron (MLP), are designed to mimic the physical characteristics of silicon PUFs under specific conditions by leveraging real datasets during the training process. The trained models are subsequently used to generate synthetic CRP datasets that preserve the physical characteristics of the real data while offering a larger number of records. This enables modeling at a larger scale and supports greater diversity across multiple devices.

Dr. Enas Abulibdeh is with the Department of Network Information Security Engineering, Princess Sumaya University for Technology (PSUT), Amman, Jordan. The author would like to sincerely thank her for her continuous support, guidance, and for being the creator of the main idea behind this work.

II. Related Work

Physical Unlclonable Functions (PUFs) are hardware-based security primitives, since hardware variations introduce randomness, modeling these PUFs will be more challenging and this research focus on modeling attacks, where deep learning models are used to predict the response of a specific challenge[2]. These deep learning models are capable of learning the underlying behavior of the hardware[2] and generate new samples of data with nearly the same statistical test scores[2], In this work instead of doing a model attack, The main objective is to generate synthetic Challenge-Response Pairs (CRPs) that emulate the real dataset that was used for training, and also to predict a response for a specific challenge achieving around 91% prediction accuraccy [2]. The main part of this job is evaluation. So, Instead of Using the 1,000,000 CRPs of the Arbiter PUF for training, We used 200,000 for testing and 800,000 for training to ensure that the model was able to learn the underlying distribution of the data.

All three models (MLP,CVAE,WGAN) successfully cloned the 4-XOR PUF. The modeled response is statistically similar to hardware real responses based on the Results shown in the NIST 800-22 test suite. At first, when we used raw 0s and 1s to train the models, it was completely failed [3]. The models got stuck at ˜50% - 59% accuracy Which means the model is learning random data and suffered from overfitting, that lead the model memorize the dataset that was improted to the model and at the same time it wasn't able to predict responses for the untrained data. But when the raw bits were converted to parity vector [4], it unscrambled the math. Confusing patterns were removed and the training became easy and effective.

Before Transformation: Initially, the models were trained using the raw binary challenge vector:

$$C = (c_1, c_2, \ldots, c_{64}), \quad c_j \in \{0, 1\} \tag{1}$$

In this case, the model receives the input directly as:

$$\text{Input} = C \tag{2}$$

Due to the internal XOR-based structure[3], [5] of the Arbiter PUF, the relationship between C and the response R becomes highly non-linear and difficult to approximate[1]. As a result, the models failed to learn data patterns.

After Transformation: To address this issue, the challenge vector is transformed into a parity vector representation [4]:

$$\phi_i = \prod_{j=i}^{64}(1 - 2c_j), \quad i = 1, 2, \ldots, 64 \tag{3}$$

$$\phi(C) = (\phi_1, \phi_2, \ldots, \phi_{64}, 1) \tag{4}$$

The model input is then defined as:

$$\text{Input} = \phi(C) \tag{5}$$

This transformation converts the original non-linear mapping into a linear form, where the response can be expressed as:

$$R = \text{sign}\left(\sum_{i=1}^{65} w_i \cdot \phi_i\right) [1], [3] \tag{6}$$

As a result, the models are able to learn the underlying behavior more effectively, leading to improved accuracy and stable training.

III. Arbiter PUF and The Generative Models

A. Arbiter Physically Unclonable Function (PUF)

This research focuses on the 4-XOR Arbiter PUF, Which is more complex of the 2-XOR Arbiter or standard Arbiter, It consists of multiple Arbiter PUFs combined using 4-XORs which makes it difficult for deep learning models to learn its underlying behavior[3].

4-XOR Arbiter PUF Dataset Description

Parameter	Value
PUF Type	4-XOR Arbiter PUF
Total CRPs	1,000,000
Training Set	800,000
Testing Set	200,000
Challenge Length	64 bits
Response Length	1 bit
Data Type	Binary (-1, 1)

B. Conditional Variational Autoencoder (CVAE)

Conditional Variational Autoencoder (CVAE) is a generative model that learns a compressed representation of data and uses it to generate new samples similar to the original distribution. Instead of memorizing the dataset, the model learns the underlying probability distribution to produce new data.

Unlike a standard Variational Autoencoder (VAE), the CVAE conditions the generation process on additional input information. In this work, the model is conditioned on the challenge to generate the corresponding response.

The architecture of the CVAE consists of the following components [6]:

- **Input:** Challenge and Response
- **Encoder:** Maps the input into a latent space by learning two parameters:
  - μ (mean of the latent distribution)
  - σ (standard deviation representing randomness)
- **Latent Vector (z):** A compressed hidden representation of the learned data
- **Decoder:** Reconstructs the response conditioned on the challenge and latent vector
- **Loss Function:**
  - Reconstruction Loss (measures prediction accuracy)
  - KL Divergence (forces the latent space to follow a normal distribution)

CVAE Training Configuration

Parameter	Value
CRPs	1,000,000
Epochs	30
Batch Size	1024
Learning Rate	0.001
Latent Dimension	32
KL Divergence Weight (β)	0.05

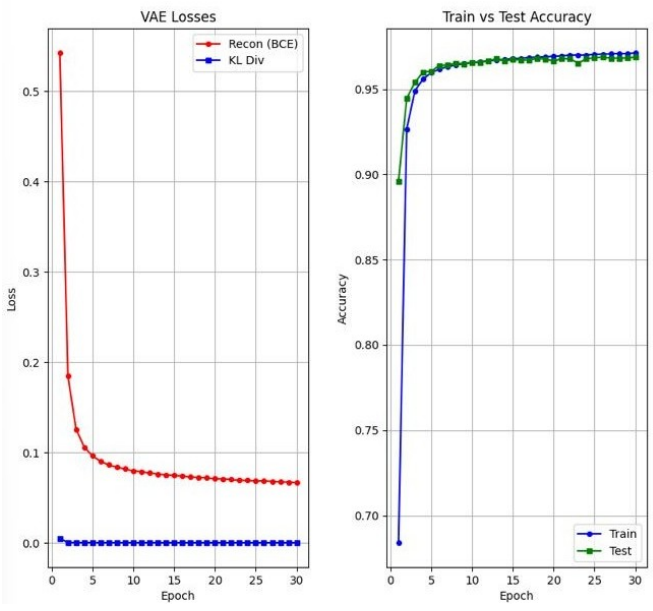


Fig. 1. The performance of CVAE training and testing.

C. Wasserstein Generative Adversarial Network (WGAN)

Wasserstein Generative Adversarial Network (WGAN) is a generative model that consists of two competing networks: a generator and a critic (discriminator).

The architecture of the WGAN consists of the following components [7]:

- **Input:** Challenge and Random Noise (z)
- **Generator:** Produces synthetic responses conditioned on the challenge
- **Critic:** Evaluates the realism of generated and real data by assigning a score
- **Loss Function:**
  - **Wasserstein Loss** (measures the distance between real and generated distributions)
  - **Gradient Penalty** (ensures stable training)

The generator produces responses conditioned on the challenge and noise that the model was trained on, while the critic evaluates and decides whether this data is real by comparing it to the real data.

To improve the quality of training, a weighted reconstruction loss was added to the generator to make the generated outputs as realistic as possible.

Although WGAN is able to learn complex data, its training process can be unstable and requires careful training configurations.

WGAN Training Configuration

Parameter	Value
CRPs	1,000,000
Epochs	30
Batch Size	1024
Learning Rate	0.0001
Z Noise Dimension	8
Reconstruction Lambda	50.0
GP Lambda	10.0
Critic Steps	5

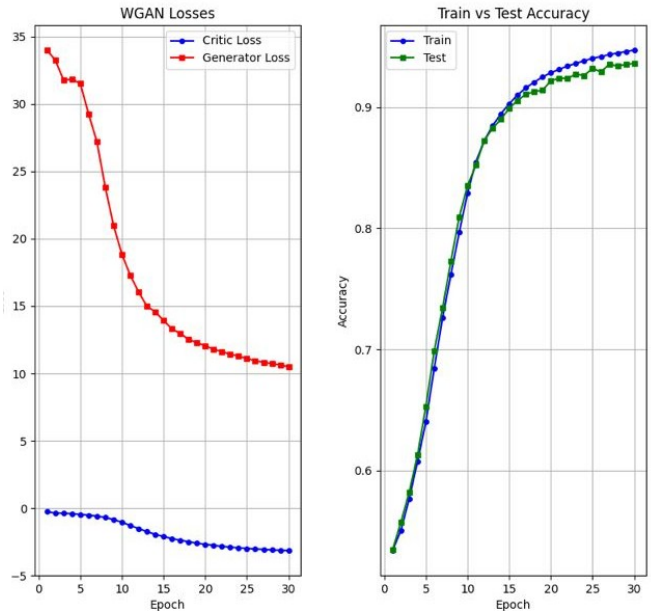


Fig. 2. The performance of WGAN training and testing.

D. Multi Layer Perceptron (MLP)

Multi-Layer Perceptron (MLP) in our objective is used to learn the mapping between the input challenges and the output responses. MLP does not learn the underlying distribution of the data, but focuses on predicting teh correct response for a specifc challenge.

The architecture of the MLP consists of the following components:

- **Input**: Challenge
- **Hidden Layers:** Multiple fully connected layers with non-linear activation functions
- **Output:** A single logit representing the predicted response
- **Loss Function:**
  - Binary Cross Entropy Loss (measures prediction error)

The model takes the challenge as input and passes it through multiple layers to produce the predicted response.

During training, the model compares the predicted output with the true response and updates its weights using backpropagation.

During evaluation, the same forward pass is applied, and a sigmoid function followed by a threshold is used to determine the final response.

The MLP model is deterministic, meaning that the same challenge always produces the same response.

Although MLP achieves high prediction accuracy, it does not capture the statistical variability of the data.

MLP Training Configuration

Parameter	Value
CRPs	1,000,000
Epochs	30
Batch Size	1024
Learning Rate	0.001
Optimizer	Adam

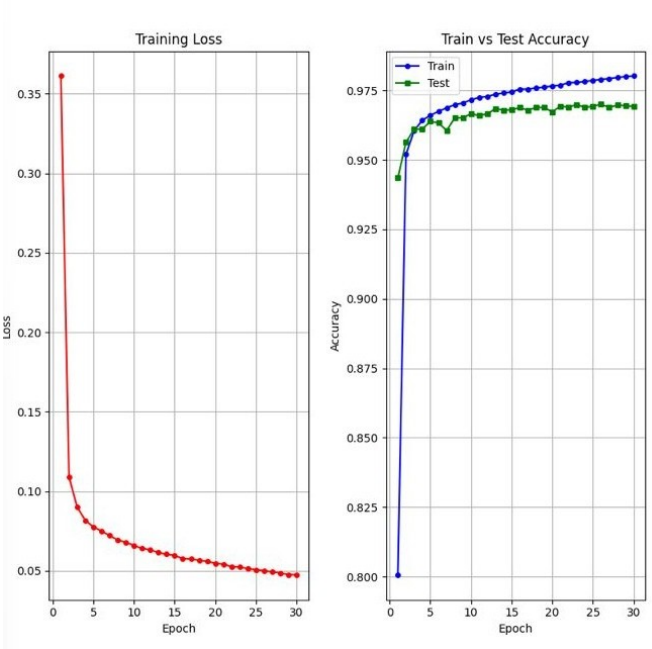


Fig. 2. The performance of MLP training and testing.

IV. Models Performance and Dataset Validation

TABLE I: Performance of Deep Learning Models on 4-XOR Arbiter PUF Dataset (800,000 CRPs for training and 200,000 for testing)

Model	Epochs	Loss	Train Acc (%)	Test Acc (%)
MLP	30	0.0475	98.02	96.9
CVAE	30	Recon: 0.0669, KL: 0.0003	97.13	96.8
WGAN	30	C: -3.1474, G: 10.4941	94.71	93.6

TABLE II: NIST NIST SP 800-22 Test Comparison: Original Dataset vs Generated Models

Test	Original	WGAN	CVAE	MLP
Frequency	0.0352 (10/10)	0.9114 (10/10)	0.2133 (10/10)	0.3505 (10/10)
Block Frequency	0.000003 (10/10)	0.5341 (10/10)	0.7399 (10/10)	0.3505 (10/10)
Cumulative Sums	0.7399 (10/10)	0.5341 (10/10)	0.7399 (9/10)	0.5341 (10/10)
Runs	0.1223 (10/10)	0.1223 (10/10)	0.7399 (10/10)	0.5341 (10/10)
Longest Run	0.2133 (10/10)	0.7399 (10/10)	0.9114 (10/10)	0.7399 (10/10)
Rank	0.3505 (10/10)	0.7399 (10/10)	0.1223 (10/10)	0.3505 (10/10)
FFT	0.0000 (0/10)	0.7399 (10/10)	0.2133 (10/10)	0.7399 (10/10)
NonOverlapping	0.0352 (10/10)	0.0089 (10/10)	0.0179 (9/10)	0.0352 (9/10)

The NIST statistical tests [2] were performed on 10 independent bitstreams, each consisting of 100,000 bits.

TABLE III: Minimum P-Value Comparison

Model	Min P-Value
Original	0.0000
WGAN	0.0089
CVAE	0.0179
MLP	0.0352

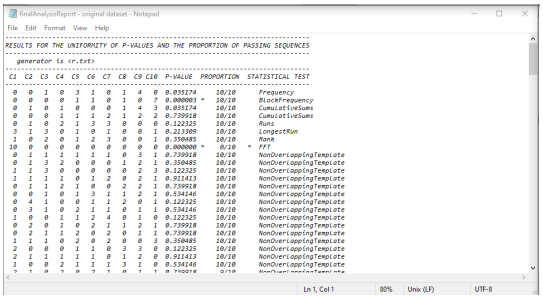
V. Conclusion

The results show that modeling PUF behaviour models gives different outcomes depending on model type. The model that achieved the highest accueacy was the MLP because it learns direct mapping between the challenge and response. So, it produces deterministic outputs. While generative models such as CVAE and WGAN focus on learning the underlying distribution of the data. CVAE showed better performance than WGAN and it was able to generate responses that was statistically close to the original dataset[2], but training was less stable in some tests.

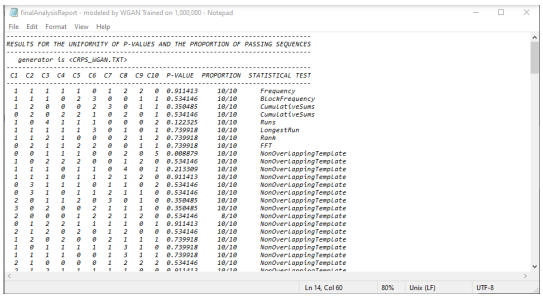
Overall, the results shows that generative models are the best for modeling PUF behavior [2], since they capture the randomness and variability of the system.

APPENDIX A

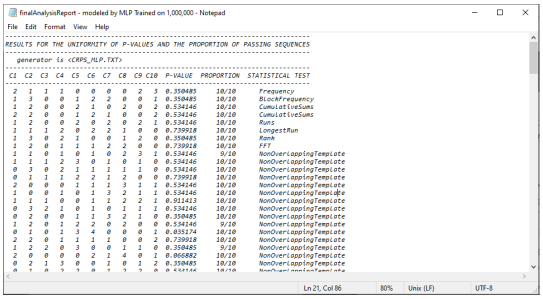
NIST SP 800-22 Test Results for Original and Generated CRPs



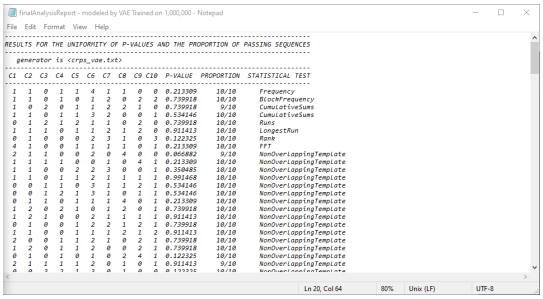
(a) Original CRPs



(b) CRPs modeled by WGAN



(c) CRPs modeled by MLP



(d) CRPs modeled by VAE

References

[1] U. Ruhrmair et al., "Modeling Attacks on Physical Unclonable Functions," in Proc. ACM CCS, 2010.

[2] N. Saadvikaa, "PUF Modeling Attacks Using Deep Learning and Machine Learning," *Engineering Proceedings*, vol. 56, no. 1, p. 187, 2023. doi: https://doi.org/10.3390/engproc2023056187
[3] F. Ganji et al., "Why Attackers Win: On the Learnability of XOR Arbiter PUFs," in Trust and Trustworthy Computing, 2015.
[4] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A Machine Learning Based Security Analysis of Strong PUFs," IACR Cryptology ePrint Archive, 2019. [Online]. Available: https://eprint.iacr.org/2019/566.pdf
[5] G. T. Becker, "The Gap Between Promise and Reality: On the Insecurity of XOR Arbiter PUFs," in CHES, 2015.
[6] M. S. Kim, S. B. Jeon, and S. Kim, "SRAM based Gaussian noise generation for post quantum cryptography," *Scientific Reports*, vol. 15, p. 43573, 2025. doi: https://doi.org/10.1038/s41598-025-27929-3
[7] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," arXiv:1701.07875, 2017.

Obeeda Al-Doulat Obeeda Al-Doulat is a student in Network Information Security at Princess Sumaya University for Technology (PSUT). His research interests include hardware security, Physical Unclonable Functions (PUFs), and machine learning applications in embedded systems.